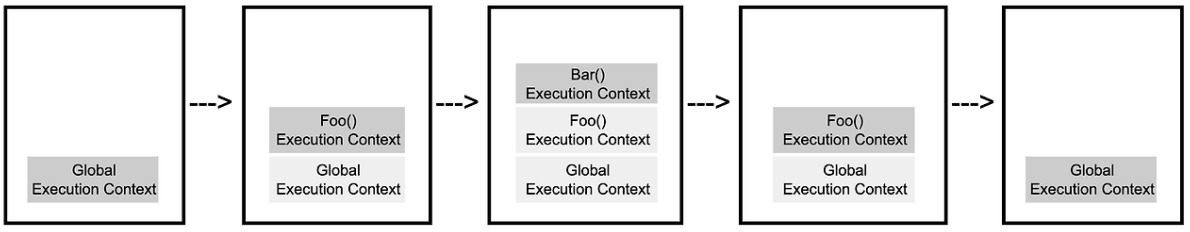
42장. 비동기 프로그래밍

42.1 동기 처리와 비동기 처리

1. 실행 컨텍스트 스택

```
const foo = () => {};
const bar = () => {};

foo();
bar();
```



2. **자바스크립트 엔진은 단 하나의 실행 컨텍스트 스택을 갖는다.**

   → 2개 이상의 함수를 동시에 실행할 수 없다는 것을 의미

3. 실행 컨텍스트 스택 최상위 요소인 "실행 중인 실행 컨텍스트"를 제외한 모든 실행 컨텍스트는 모두 실행 대기 중인 태스크(task)

4. 자바스크립트 엔진은 한 번에 하나의 태스크만 실행할 수 있는 **싱글 스레드(single thread)** 방식
   - 처리에 시간이 걸리는 태스크를 실행하는 경우 **블로킹(작업 중단)** 발생

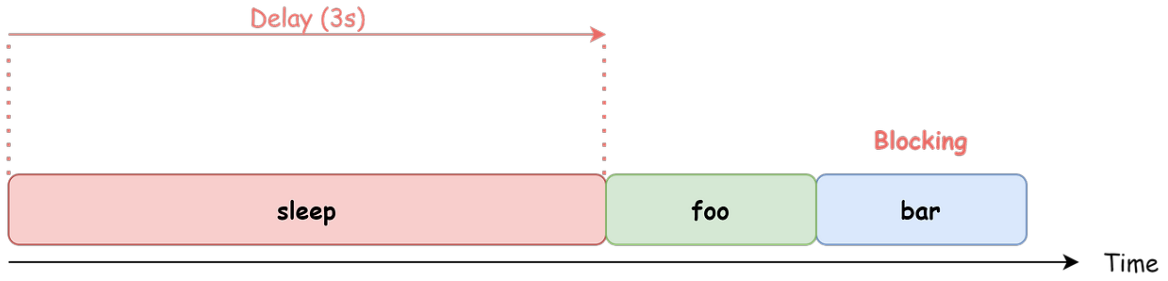
동기(synchronous) 처리

장점

- 순서대로 하나씩 처리하느라 실행 순서 보장

단점

- 앞선 태스크 종료때까지 이후 태스크들이 블로킹 됨



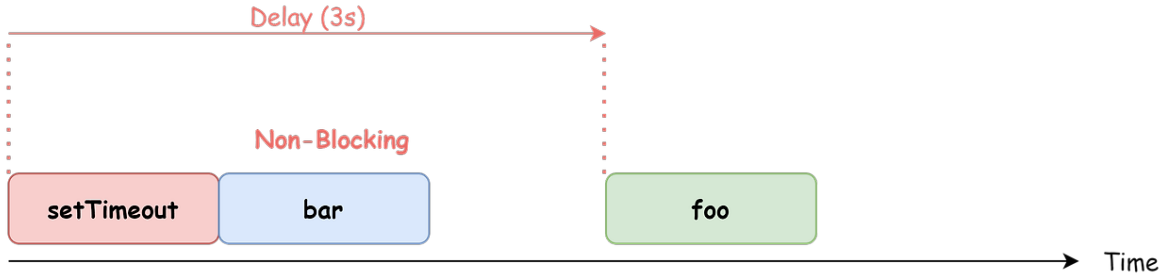
비동기(asynchronous) 처리

장점

- 현재 실행 중인 태스크가 종료되지 않은 상태여도 다음 태스크를 곧바로 실행 → 블로킹 발생 X

단점

- 태스크의 실행 순서 보장 X



6. **타이머 함수 setTimeout, setInterval, HTTP 요청, 이벤트 핸들러 : 비동기 처리 방식으로 동작**

42.2 이벤트 루프와 태스트 큐

1. **이벤트 루프(event loop)** : 자바스크립트의 동시성(concurrency)을 지원하는 것

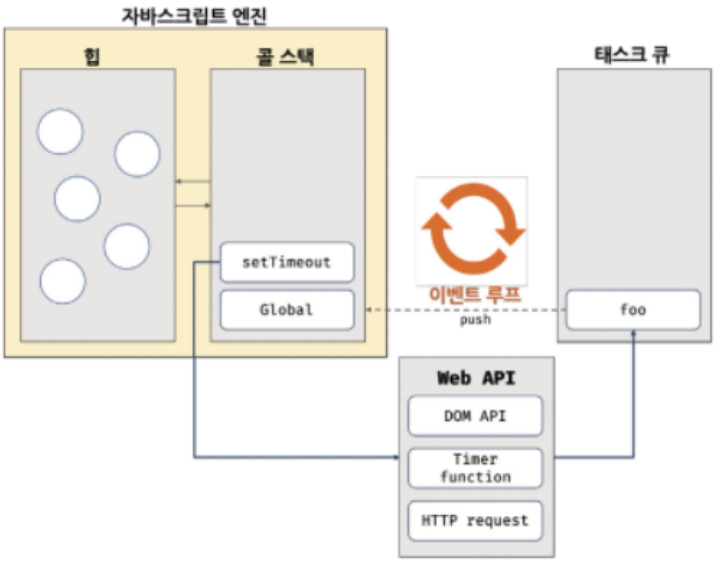


그림 42-4 이벤트 루프와 브라우저 환경

2. 이벤트 루프는 브라우저에 내장되어 있다.구글의 V8 자바스크립트 엔진을 비롯한 대부분의 자바스크립트 엔진은 크게 2개의 영역으로 구분된다

- **콜스택(call stack)** : 실행 컨텍스트 스택이 바로 콜 스택.
- **힙(heap)** : 객체가 저장되는 메모리 공간.

3. 비동기 함수 setTimeout의 평가와 실행은 자바스크립트 엔진이 담당하지만 호출 스케줄링을위한 타이머 설정과 콜백 함수의 등록은 브라우저 또는 Node.js가 담당한다.

   이를 위해 브라우저 환경은 태스크 큐와 이벤트 루프를 제공한다.

- **태스크 큐(tast queue/event queue/callback queue)**

  : setTime과 같은 비동기 함수의 콜백 함수 or 이벤트 핸들러가 일시적으로 보관되는 영역

- **이벤트 루프(event loop)**

  : 이벤트 루프는 콜 스택에 현재 실행 중인 실행 컨텍스트가 있는지, 그리고 태스트 큐에 대기 중인 함수(콜백 함수, 이벤트 핸들러 등)가 있는지 반복해서 확인. **만약 콜 스택이 비어있고 태스크 큐에 대기 중인 함수가 있다면 이벤트 루프는 순차적으로(FIFO) 태스크 큐에 대기 중인 함수를 콜 스택으로 이동시킴.**

4. [예제 42-04]

```
function foo() {
  console.log('foo');
}
```

```
function bar() {
  console.log('bar');

setTimeout(foo, 0); // 0초(실제는 4ms) 후에 foo 함수가 호출된다.
bar();
```

1. 전역 코드가 평가되어 전역 실행 컨텍스트가 생성되고 콜 스택에 푸시된다.

2. 전역 코드가 실행되기 시작하여 setTimeout 함수가 호출된다. 이따 setTimeout 함수의 실행 컨텍스트가 생성되고 콜 스택에 푸시되어 현재 실행 중인 실행 컨텍스트가 된다. 브라우저의 Web API인 타이머 함수도 함수이므로 함수 실행 컨텍스트를 생성한다.

3. setTimeout 함수가 실행되면 콜백 함수를 호출 스케줄링하고 종료되어 콜 스택에서 제거된다. 이때 호출 스케줄링, 즉 타이머 설정과 타이머가 만료되면 콜백 함수를 태스크 큐에 푸시하는 것은 브라우저 역할이다.

4. 브라우저가 수행하는 4-1과 자바스크립트 엔진이 수행하는 4-2는 병행 처리된다.

    4-1. 브라우저는 타이머를 설정하고 타이머의 만료를 기다린다. 이후 타이머가 만료되면 콜백 함수 foo가 태스크 큐에 푸시된다. 예제의 경우 지연 시간(delay)가 0이지만 최소 지연 시간 4ms가 지정된다. 따라서 4ms 후에 콜백 함수 foo가 태스크 큐에 푸시되어 대기하게 된다. 이 처리 또한 자바스크립트 엔진이 아니라 브라우저가 수행한다.

    4-2. bar 함수가 호출되어 bar 함수의 실행 컨텍스트가 생성되고 콜 스택에 푸시되어 현재 실행 중인 실행 컨텍스트가 된다. 이후 bar 함수가 종료되어 콜 스택에서 제거된다. 이때 foo 함수는 4ms 시간이 경과 했더라도 아직 태스크 큐에서 대기 중이다. (아직 전역 컨텍스트가 남아 있기 때문에)

5. 전역 코드 실행이 종료되고 전역 실행 컨텍스트가 콜 스택에서 제거된다. 이로서 콜 스택에는 아무것도 없다.

6. 이벤트 루프에 의해 콜 스택이 비어 있음이 감지되고 태스크 큐에서 대기 중인 콜백 함수 foo가 이벤트 루프에 의해 콜 스택에 푸시되어 실행된다. 이후 foo 함수가 종료되어 콜 스택에서 제거됨.

5. 이처럼 **비동기 함수인 setTimeout의 콜백 함수는 태스크 큐에 푸시되어 대기하다가 콜 스택이 비게 되면,즉, 호출된 함수가 모두 종료하면 그때 콜 스택에 푸시되어 실행된다.**

   자바스크립트는 자바스크립트 엔진에 의해 싱글 스레드 방식으로 동작한다.브라우저는 멀티 스레드로 동작한다.